第5章 TCP/IP网络配置

在本章中，我们将引导大家在自己的机器上实施设置 TCP/IP联网的全部必要过程。首先从IP地址的分配入手，再谈谈 TCP/IP网络接口的配置，随后针对网络安装过程中容易出现的几个问题，为大家介绍几个方便易用的工具。

本章涵括的任务大多只须执行一次。然后，只有在将新系统添加到自己的网络中时，或重新全盘配置你的整个系统时，才有必要接触到大量的配置文件。但是用于配置 TCP/IP的命令中，有些是每次启动系统时，都必须执行的。通常从 system/etc/rc脚本中调用这些命令。

注意 系统初始化脚本有两大主要派别：BSD风格和SysV init风格。Res Hat Linux采用的是修正过的SysV风格的初始化进程。这里的rc脚本和命名约定引用的是BSD风格的初始化进程。

通常，初始化进程的网络专有部分包含在一个名为 rc.net或rc.inet的脚本中。有时，还可能看见两个脚本，其名分别为 rc.inet1和rc.inet2，前者初始化联网的内核部分，后者开始基本的联网服务和应用。下面的示例中，我们采用的是后一个脚本。

rc.inet1脚本执行的动作和应用将留在后续章节中讨论。本章结束之时，大家应该建立起一序列的命令，这些命令完全能够完成 TCP/IP联网的配置工作。然后，再用 rc.inet1中的任何一个示例命令来替换它们，以保证 rc.inet1是在系统启动时执行的，并重新启动你的机器。联网用的rc脚本会给你一个较好的示例。

5.1 proc文件系统的设置

Net-2发布的版本中，有些配置工具依赖于 proc文件系统和内核进行通信（在内核 2.2.x版本中，联网代码被说成是 Net4）。这是一个接口，它允许通过一个类似文件系统的机制，访问内核运行时信息。装入时，可像对待其他所有文件系统一样，列出其所有文件，或显示其中的所有内容。典型的项目有 loadavg文件（其中包含系统载入平均数）或 meminfo（显示当前的核心内存和交换区的用法）。

所以，联网代码增加网络目录。其中包含大量的文件，分别显示内核 ARP表、TCP连接状态和路由信息表等等。许多网络管理工具都是从这类文件中获取相关信息的。

proc文件系统（或procfs）通常在系统启动时被装入 /proc。最好的办法是在 /etc/fstab中增加下面的代码：

```
# procfs mont point:
none            /proc       proc        defaults
```

然后，再从/etc/rc脚本中执行mount/proc。

如今，procfs通常采用默认设置进入内核。如果 procfs不在你的内核中，你就会得到这样一条消息：“ mount:fs type procfs not supported by kernel ”（内核不支持fs类型的procfs）。之后，

必须重新编译内核，并在要求回答是否需要 procfs支持时，回答“ Yes ”（是）。

5.2 二进制文件的安装

如果你采用的是预封装的工具，其中多半包含有主要的连网应用程序和实用程序以及一个混然一体的示例文件集。必须获得并安装这些新实用程序的唯一的条件是你安装了一个最新版本的内核。由于内核连网层时时可能发生变化，所以有必要经常性地更新自己的基本配置工具。至少须重新编译内核，但有时，还会要求你必须有最新的二进制文件集。这些文件一般随内核一起发放，封装在一个名为 net-XXX.tar.gz的归档文件中， XXX代表的是版本号。和-1.0对应的发布是0.32b，本书完稿之时，最新版本的内核（ 1.1.12）需要的是0.32d。

如果你想自行编译和安装标准的 TCP/IP网络，可以从大多数 FTP服务器那里获得源代码。这些源代码都是在 Net-BSD或其他源代码基础上，经过许多人的修订而成的。其他的应用程序，比如Xmosaic、Xarchie、 Gopher和IRC客户机，则必须单独从特定的地点获得。如按照指导去做，大多数程序都是行得通的。

5.3 另一个例子

从现在开始，我们将讨论一个更为实际的例子，它比 Groucho Marx大学校园网稍微简单一些。它就是Virtual Brewery。这是一家酿造啤酒的小公司。为了更有效地对自己的业务进行管理，老板想把各台机器连接起来，所有的机器都是新的。

同一层楼上，穿过大厅，就是Virtual Winery，这家公司的性质和Brewery差不多。他们各自运行一个以太网。很自然，只要可行的话，两家公司都想链接到对方的网络上。作为第一步，先在两个子网中间设置一个转发数据报的网关主机。然后，设法通过 UUCP链接到公司外面，以便收发新闻和电子邮件。以后，肯定是设置 SLIP链接接入因特网。

5.4 设置主机名

多数情况下（非绝对的），网络应用程序依赖于本地主机名（已被设为一个恰当的值）。这是在系统启动过程期间，利用 hostname命令来完成的。要设置主机名，就应该像下面这样调用它

```
# hostname name
```

通常用没有域名的不合格主机名来进行试验。比如， Virtual Brewery公司的主机可以是vale.vbrew.com、vlager.vbrew.com等。它们是这些主机的正式主机名，即完整资格域名。其本地主机名只能是公司名的前几个字母，比如 vale。但是，由于本地主机名常用于查找主机的IP地址，所以必须确保解析器库能够查找主机的 IP地址。这通常意味着必须在 /etc/hosts（参见下文）内输入主机名。

有人建议利用 domain name（域名）命令，将内核的域名概念设置为 FQDN的其他部分。通过这一方式，可以把主机名和域名的输出组合起来，获得一个 FQDN。但是，这也不能保证百分之百的正确。域名一般用于设置主机的 NIS（网络信息系统）域，这个域完全不同于你的主机所属的DNS。关于NIS的详情，请参见第9章。

5.5 分配IP地址

在自己的主机上配置联网软件，进行单机操作时（例如，能够运行 INN netnews软件），

完全可忽略本小节，因为你只需要回送接口的 IP地址，而这个接口的地址始终都是 127.0.0.1。

但面对真正的网络时（比如以太网），事情就没那么简单了。如果想将你的主机连接到一个现有网络，必须要求该网络的管理员为你分配一个 IP地址。在自行设置网络时，你必须像下面这样，给自己分配一个地址。

一个本地网络内的主机通常共享同一个逻辑 IP网络的地址。因此，你必须分配一个 IP网络地址。如果你手中有若干个物理网络，要么为它们分配不同的网络编号，要么利用子网技术，把自己的IP地址范围分成若干个子网。

如果你的网络没有接入因特网，便可以根据自己的喜好，自由选择网络地址，只要它是合法的。同时，必须保证从 A、B或C类地址中选择。但是，如果你打算不久将接入因特网，现在就应该想办法获得正式的 IP地址。然后，最好要求你的网络服务提供商为你提供相关帮助。如果想获得有朝一日用于因特网的网络编号，要向 hostmaster@internic.net请求一个“ 网络地址申请表 ”。

要想同时操作若干个以太网（或其他网络，只要驱动程序允许），就必须将自己的网络分成若干个子网。注意，只有你手中的广播网络不止一个时，才需要子网。点到点链接不能算作广播网络。例如，如果你有一个以太网和一个以上的 SLIP链接（接到外部世界），就不必将自己的网络分成若干个子网。

现在回到我们的示例上来。 brewery的网管采用的是NIC的B类网络编号，分配到的地址是191.72.0.0。为了能容纳两个以太网，她决定采用8位主机部分作为其增加的子网位。另8位用于主机部分，每个子网上允许接纳 254台主机。然后，她将子网编号 1分配给brewery，2分配给winery。因此，它们各自的网络地址分别是191.72.1.0和191.72.2.0。子网掩码是255.255.255.0。

作为两个网络间网关的 vlager，在两个网络上分到的主机编号都是 1，但它分到的IP地址分别是191.72.1.1和191.72.2.1。下面的示例展示了两个子网和网关。

```
#
# Hosts file for Virtual Brewery/Virtual Winery
#
# IP            local       fully qualified domain name
#
127.0.0.1       localhost
#
191.72.1.1      vlager      vlager.vbrew.com
191.72.1.1      vlager-if1
191.72.1.2      vstout      vstout.vbrew.com
191.72.1.3      vale        vale.vbrew.com
#
191.72.2.1      vlager-if2
191.72.2.2      vbeaujolais  vbeaujolais.vbrew.com
191.72.2.3      vbardolino   vbardolino.vbrew.com
191.72.2.4      vchianti     vchianti.vbrew.com
```

注意，这个示例中，采用的是B类网络，目的是为了简明扼要；其实， C类网络更为常见。有了新联网代码，子网的划分就不再受到字节边界的限制，所以C类网络也可分成若干个子网。例如，你可将两位的主机编号用于网络掩码，从而网络可分为四个子网，每个子网上可以有 64台主机(每个子网上的最后一个编号是为广播地址保留的，所以事实上每个子网只有63台主机)。

5.6 编写主机和网络文件

在将自己的网络分成若干个子网后，应该利用 /etc/hosts文件，进行简单的主机名解析了。如果不用DNS和NIS来进行地址解析，就必须将所有的主机信息放在一个 hosts文件(主机)内。

即使你想在普通操作期间运行 DNS或NIS，也应该先有/etc/hosts文件内的某些子网的所有主机名。比如在无网络接口运行的情况下 (引导时)，你想有某类主机名解析，那么，这样不仅很方便，还允许你采用 rc.inet脚本内的象征性的主机名。因此，在更改 IP地址时，只需将一份更新过的主机文件复制到所有的机器中，再重新启动即可，而不是重新逐个编辑数量庞大的rc文件。通常情况下，可把所有的本地主机名和地址放入主机文件内，如果需要，还可将网关和NIS服务器添加在内。(只有采用Peter Eriksson编写的NYS时，才需要NIS服务器的地址。其他NIS实施位于它们自己的服务器上，只能在运行时用 ypbind获得。)

与此同时，初始测试期间，你应该保证自己的解析器只采用主机文件内的信息。在使用自己的DNS和NIS软件时，可能会利用一些会产生怪异结果的示例文件。查询主机 IP地址时，如果要所有应用程序专用 /etc/hosts文件，必须对/etc/host.conf文件进行编辑。在以关键字 order开头的所有代码行前加上一个“ # ”号，就可以批注出所有的行，并插入下面这行

```
order hosts
```

注意 如何配置解析器库，请参见第6章。

主机文件中每行都有一个条目，由 IP地址、主机名和供选择的主机名别名清单组成。这些字段用空格或标号隔开，而且地址字段必须在第一列内。跟在“ # ”号后面的被视作批注，可忽略。

主机名要么已完全通过验证，要么关联在本地域内。以 vale为例，通常输入的是其完整资格名， vale.vbrew.com和主机文件内的 vale本身，所以它就有两个名字，一个是正式主机名，另一个是较短的本地名。

在我们上面的示例中， Virtual Brewery的主机文件像下面这样。其中包括两个特殊的主机名：vlager-if1和vlager-if2，它们为vlager网关上用的两个接口提供了地址。

正如主机的IP地址一样，有时人们肯定想用一个特殊好记的名字来代表某个网络。所以，主机文件还有一个如影随行的“ 搭档 ”—— /etc/networks，该文件指明网络名和网络编号之间的对应关系。

```
#
# Hosts file for Virtual Brewery/Virtual Winery
#
# IP            local       fully qualified domain name
#
127.0.0.1       localhost
#
191.72.1.1      vlager      vlager.vbrew.com
191.72.1.1      vlager-if1
191.72.1.2      vstout      vstout.vbrew.com
191.72.1.3      vale        vale.vbrew.com
#
191.72.2.1      vlager-if2
191.72.2.2      vbeaujolais  vbeaujolais.vbrew.com
```

```
191.72.2.3      vbardolino    vbardolino.vbrew.com
191.72.2.4      vchianti      vchianti.vbrew.com
```

注意 Res Hat Linux中没有/etc/networks文件。

我们在Virtual Brewery安装了一个networks（网络）文件，如下所示：

```
# /etc/networks for the Virtual Brewery
brew-net 191.72.1.0
wine-net 191.72.2.0
```

注意 网络文件中的网络名不得与主机文件内的主机名冲突，否则，会令某些程序产生怪异的结果。

5.7 IP接口配置

像前一章描述的那样设置好硬件之后，接下来的任务是让内核连网软件知道它们的存在。配置网络接口，并初始化路由表的命令有两条。配置任务通常是在系统启动时，从 rc.inet1脚本开始进行的。采用的基本工具叫作 ifconfig（if代表接口）和route，即接口配置和路由。

ifconfig用于令接口能够被内核联网层访问。这涉及到 IP地址和其他参数的分配，接口的激活（有时也称作启用）。这里的激活意思是内核将通过该接口收发 IP数据报。要想启用接口，最简单的方式是利用下面的代码调用它

```
ifconfig interface ip-address
```

上面的代码将ip-address分给接口，就激活了它。 其他所有参数都采用默认值。例如，默认的子网掩码衍生于 IP地址的网络类，比如代表 B类地址的255.255.0.0（ifconfig工具的详情，将在本章最后进行解释）。

第二个工具是route，它用于在内核路由表内增加或删除路由。可以这样调用它：

```
route [add|del] target
```

add和del这两个参数用于判断是增加还是删除路由。

5.7.1 回送接口

首先激活的接口是回送接口：

```
# ifconfig l0 127.0.0.1
```

有时，还可以看到本地主机用伪主机名代替了 IP地址的情况。 ifconfig将查找主机文件内的主机名，其中一个条目将该主机声明为 127.0.0.1的主机名：

```
# Sample /etc/hosts entry for localhost
localhost 127.0.0.1
```

如果要查看接口的配置情况，将接口名作为 ifconfig的参数，调用它即可：

```
$ ifconfig lo
lo        Link encap Local Loopback
inet addr 127.0.0.1  Bcast [NONE SET]  Mask 255.0.0.0
UP BROADCAST LOOPBACK RUNNING  MTU 2000  Metric 1
RX packets 0 errors 0 dropped 0 overrun 0
TX packets 0 errors 0 dropped 0 overrun 0
```

由此可知，回送接口已经分到一个网络掩码 —— 255.0.0.0，这是因为127.0.0.1是A类地址。这个接口也没有设置广播地址，因为对回送接口而言，一般没有多大用处。但是，如果你在

自己的主机上运行rwhod程序，就必须为回送设备设置广播地址，只有这样，才能保证 rwho正常运行。关于广播地址的设置，如下所示。

现在，你几乎可以运行你的网络了，但还缺一项，即路由表中的一条，它告诉 IP可以使用这个接口作为去往目标127.0.0.1的路由，可通过键入下行命令完成：

```
# route add 127.0.0.1
```

再次提醒大家注意，可以用本地主机名来代替 IP地址。

接下来，应该利用ping对所有的配置进行校验。ping相当于一种连网用的声纳设备，用于验证所给地址是否能够抵达，向它发送一个数据报后，稍隔一会，就会返回。一个 ping所花的时间一般称为一个周期（round-trip）。

```
# ping localhost
PING localhost (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp seq=0 ttl=32 time=1 ms
64 bytes from 127.0.0.1: icmp seq=1 ttl=32 time=0 ms
64 bytes from 127.0.0.1: icmp seq=2 ttl=32 time=0 ms
^C

— localhost ping statistics —
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0/0/1 ms
```

像上面这样调用 ping时，它会永远不停地发送数据包，除非用户将其中断。上面的 ^C标出了我们按Ctrl+C中断校验的地方。

上面的示例展示了发到127.0.0.1的数据包传递无误，返回给ping的应答几乎是同步的。这表明你已成功设置了第一个网络接口。

如果ping的输出和上面的示例有出入，你就麻烦了。查看有些文件的安装是否有误。再看看你使用的ifconfig和route二进制文件是否与自己运行的内核兼容，最重要的是看内核与已启用的连网代码是否兼容（可从/proc/net目录得知）。如果得到的错误消息是这样的：“网络不可抵达”，说明你可能把route命令搞错了。一定要保证该命令中采用的地址和 ifconfig中的地址是一样的。

通过上面的步骤，完全足以在一台独立主机上使用连网应用程序了。在 rc.inet1增加上面的代码行，并保证从/etc.rc执行这两个rc.inet脚本之后，可能需要重新启动机器，试着运行各种应用程序。例如，“telnet localhost” 将建立一条通向你的主机的telnet链接，并给出一个登录提示。

然而，回送接口不仅用作网络丛书的示例，还用作开发过程中的测试床，但最常见的仍然是供有些应用程序用于普通操作。例如，基于 RPC的所有应用程序都利用回送接口，随portmapper程序一起在启动时注册自己。因此，这个接口总应该配置好，不管你的机器是否接入网络。

5.7.2 以太网接口

以太网接口的配置和回送接口的配置大致相同，唯一例外的是使用子网技术时，它要求的参数比后者要多一些。

在Virtual Brewery示例中，我们已经把起初是 B类网络的IP网络分成了若干个 C类子网。

要使接口能够对此进行识别，需像下面这样调用 ifconfig：

```
# ifconfig eth0 vstout netmask 255.255.255.255.0
```

这样，便为eth0接口分配了一个 vstout IP地址（191.72.1.2）。如果我们省略了网络掩码，ifconfig就会根据IP网络的类别推断出网络掩码是什么，而且这个网络掩码将是 255.255.0.0。现在，快速查看展示了下面的内容：

```
# ifconfig eth0
eth0      Link encap 10Mps Ethernet HWaddr  00:00:C0:90:B3:42
inet addr 191.72.1.2 Bcast 191.72.1.255 Mask 255.255.255.0
UP BROADCAST RUNNING  MTU 1500  Metric 1
RX packets 0 errors 0 dropped 0 overrun 0
TX packets 0 errors 0 dropped 0 overrun 0
```

由上可知，ifconfig自动将广播地址（上面的Bcast字段）设为普通值，即设置了所有主机位的主机网络编号。同时，消息传输单元（内核必须为该接口生成的以太帧最大字节数）必须设为最大，即1 500个字节。所有这些值都优先于后面所讲的特殊选项。

接下来的情形和回送接口类似，同样必须安装一个路由条目，向内核报告通过 eth0接口能够抵达的网络有哪些。就Virtual Brewery而言，应将route当作下面的代码行进行调用：

```
#route add -net 191.72.1.0
```

最初，这显得有点奇怪，因为真的不清楚 route如何检测出路由通过哪个接口。然而，这其实很简单：内核检查至今为止已配置好的所有接口，把目的地址 (此处为191.72.1.0)和接口地址的网络部分比较。唯一匹配的接口是 etho.

现在看看-net选项是干什么的。这被选用，因为 route即可以处理通向网络又可处理通向单个主机的路由。当给出点十进制地址后，它通过观察主机部分得出它是一个网络还是一个主机。如果地址的主机部分是 0，route假定它代表网络，否则它被视为主机地址。因此 route认为191.72.1.0是主机地址而非网络地址，因为它不知道我们使用了子网。所以我们必须使用 -net标记显式告诉它，此地址代表网络。

当然，上面的route命令敲起来很乏味，而且易于拼错。更方便的方法是使用我们在letc网络中定义的网络名。这使该命令更易读，此时-net标记可省略，因为route知道191.72.1.0代表网络。

```
# route add brew-net
```

现在，基本的配置步骤已告一段落，我们希望你的以太网接口能够正常运行。从你自己的以太网中选出一台主机，比如 vlager，并键入

```
# ping vlager
PING vlager: 64 byte packets
64 bytes from 191.72.1.1: icmp seq=0. time=11. ms
64 bytes from 191.72.1.1: icmp seq=1. time=7. ms
64 bytes from 191.72.1.1: icmp seq=2. time=12. ms
64 bytes from 191.72.1.1: icmp seq=3. time=3. ms
^C

—vstout.vbrew.com PING Statistics—
4 packets transmitted, 4 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 3/8/12
```

如果没有看到类似的输出，很显然，有什么被中断了。如果包丢失率异乎寻常的高，通常意味着硬件方面出问题了，比如终端坏了，或根本没有终端等等。如果根本不能接收数据包，就应该利用 netstat检查一些接口配置。 ifconfig显示出来的包特性将告诉你是否在某个接

口上发送了数据包。如果还访问了远程主机，也应该对这台远程主机及其接口特性进行检查。通过这种方式，便可准确地判断数据包是从哪里开始丢失的。另外，还应该利用 route显示路由信息，以了解两台主机的路由条目是否正确。如果在调用 route时，不带任何参数，就会在屏幕上输出路由信息（-n选项只能令其以点分四段式打印地址，而不是主机名）：

```
# route -n
Kernel routing table
Destination     Gateway         Genmask         Flags   Metric  Ref    Use
127.0.0.1       *               255.255.255.255 UH      1       0
127.72.1.0      *               255.255.255.0   U       1       0
```

这些字段的详细解释如下。 Flag列中包含一列针对每个接口的标记。 U始终是为活动接口设置的，H的意思是目标地址表示一台主机。如果 H标记是为准备用于网络路由的路由设置的，就必须采用带上-net选项的route命令。要验证你键入的路由真正能用，须检查两次调用 ping期间，倒数第二列中的Use字段是否有所增加。

5.7.3 通过网关的路由

前一小节，我们只介绍了在一个单独的以太网上设置主机的情况。但更为常见的是，通过若干个网关连接起来的网络群。这些网关虽然只链接了两个或两个以上的以太网，但也可以链接到外部网络，比如说因特网。要想使用网关提供的服务，必须为网络层提供给额外的路由信息。

例如， Virtual Brewery和Virtual Winery各自的以太网通过 vlager之类的网关链接在一起。假设vlager已经配置就绪，我们只须在 vstout的路由表内增加另一个条目即可，这个路由表将告诉内核它能够通过 vlager，抵达Winery网络上的所有主机。相应的 route调用如下所示； gw关键字告诉内核，下一个参数表示一个网关：

```
# route add wine-net gw vlager
```

当然，对于你希望与之通信的 Winery网络上的任何一台主机，都必须有相应的用于Brewery网络的路由条目，否则，就只能将数据从 vstout发送到vbardolino，但后者返回的所有应答都将进入一个很大的位桶（bit bucket）。

这个示例只说明了两个独立以太网之间，用于交换数据包的一个网关。现在，我们假设vlager还接入了因特网（比方说通过一条新增的 SLIP链接接入）。如果我们希望数据报能发送到非Brewery的任何一个网络，就应该把数据报交给 vlager。这是通过令其作为vstout的默认网关来完成的：

```
# route add default gw vlager
```

网络名的默认设置是0.0.0.0的速写，表示默认路由。不必将这个网络名加入 /etc/networks文件中，因为它是内置入默认路由中的。

如果在一个或多个网络之后，对一台主机进行 ping测试时，发现包丢失率很高，这通常意味着网络阻塞得相当厉害。包的丢失归咎于技术上的落后，或临时性的网络超载，因此主机产生延迟，甚至丢弃进入的数据报。

5.7.4 网关的配置

要配置一台机器，使其负责两个以太网间的包交换，是非常容易的一件事。假设以 vlager为例，它配有两张以太网卡，每张卡都连接到其中一个网络上。你只须为它们分配 IP地址，并

对这两个接口进行配置即可。

注意 Linux-2.0.x系列中，默认情况下的包转发是取消了的。所以可在运行时启用包的转发：# echo 1 > /proc/sys/net/ipv4/ip_forward。

像下面这样，在主机文件内增加关于这两个接口的信息是相当有用的，关于这两个接口，我们还有更容易记的名字：

```
191.72.1.1    vlager    vlager.vbrew.com
191.72.1.1    vlager-if1
191.72.2.1    vlager-if2
```

下面的命令用于设置这两个接口：

```
# ifconfig eth0 vlager-if1
# ifconfig eth1 vlager-if2
# route add brew-net
# route add wine-net
```

5.7.5 PLIP接口

与利用以太网连接相比，利用 PLIP链接连接两台计算机的情况稍有不同。后者是所谓的点到点链接，因为相对广播网络而言，它只涉及到两台主机（即节点）。

我们以Virtual Brewery公司的一名员工手中的膝上型计算机为例，该公司的网络通过 PLIP与vlager连接。这台膝上型计算机本身名为 vlite，而且只有一个并行端口。在启动时，这个端口被注册为plip1。要激活这条链接，必须利用下面的命令，配置 plip1接口：

```
# ifconfig plip1 vlite pointtopoint vlager
# route add default gw vlager
```

注意 pointtopoint不是排版错误，它真的就是这样拼写的。

第一个命令行告诉内核这是一条点到点链接，其远程端的地址是 vlager，配置了plip1接口。第二个命令行将 vlager作为网关，安装了默认路由。在 vlager上，还有必要利用一个类似的ifconfig命令激活这条链接（不必调用 route）：

```
# ifconfig plip1 vlager pointtopoint vlite
```

有趣的是：vlager上的plip1接口居然不必有其独立的IP地址，而是给定的191.72.1.1这个地址。但是，应该注意，只有在完全配置好以太网路由表之后，才能配置 PLIP或SLIP链接。不然的话，在使用有些老版本的内核时，会在配置点到点链接时中断。

膝上型计算机的路由配置之后，转向 Brewery网络的路由配置；还没有为 Brewery网络上的任何一台主机配置到vlite的路由呢！特别麻烦的一种方法是在每台主机的路由表上增加特定的路由，这些路由表将vlager指定为通向vlite的网关：

```
# route add vlite gw vlager
```

因此，在面对临时性的路由时，最好采用动态路由信息。其一是利用 gated（一种路由程序），首先在网络中的各台主机上安装这个程序，以便动态分发路由信息。但是，最简单的方法是利用代理ARP。有了代理ARP，只要一有查询vlite的ARP，vlager就会发送自己的以太网

地址，作出相应的应答。这样的后果是所有发向 vlite的包都会流入vlager，再由vlager将它们转发到膝上型计算机。我们将在随后的PLIP小节，为大家谈谈代理ARP。

将来的Net-3中，将包含一个名为plipconfig的工具，允许你设置打印机端口的IRQ。稍后，可能还有更为常用的ifconfig命令取而代之。

5.7.6 SLIP和PPP接口

尽管SLIP和PPP链接只是诸如PLIP之类的简单点到点链接，但其内容还是相当纷繁的。通常，SLIP连接涉及的步骤有：通过modem，拨打远程站点，将串行线路设置为SLIP模式等等。PPP的用法与此类似。设置SLIP或PPP链接需要哪些工具呢，详情参见第6和第7章。

5.7.7 伪接口

伪接口其实是个舶来品，但它的确有用。它给单机和其 IP网络连接是一条拨号链接的计算机带来诸多的好处。事实上，后者大多数时间也是单机。

单机的局限在于：它们只有一个单一的网络设备是活动的，也就是说仅有一个回送设备，这个设备的地址始终是 127.0.0.1。但是，我们偶尔需要向本地主机的正式 IP地址发送数据。以膝上型计算机vlite为例。这时，它已经取消了所有的网络连接。 vlite上的一个应用程序可能打算想向一台机器上的另一个应用程序发送某些数据。怎么办？在 vlite的/etc/hosts1文件内查找IP地址191.72.1.65，如此一来，应用程序便开始试着向这个地址发送数据。由于回送接口是当前本机上唯一的活动接口，所以内核无法判断这个地址引用的其实就是它本身！其结果是，内核将数据报丢弃，并向应用程序返回一个错误。

因此，伪设备应运而生。只须充当回送接口的替身，便解决可上面的问题。在 vlite这个示例中，只须给它一个 191.72.1.65这样的地址，并增加一条指向它的主机路由即可。每个目标为191.72.1.65的数据报都会得以在本地传送。其正确的调用方法是：

```
# ifconfig dummy vlite
# route add vlite
```

5.8 ifconfig详解

ifconfig的参数非常之多，远远超过了以前所讲过的那些。其普通调用结构是：

```
inconfig interface [[-net|-host] address [parameters]]
```

“ interface ”指接口名，“ address ”指准备分配给这个接口的IP地址。它即可以是点分四段式的IP地址，又可以是ifconfig在/etc/hosts和/etc/networks文件内找出的接口名。 -net和-host选项强制ifconfig将这个地址视作网络编号或主机地址。

如果ifconfig在调用时只有接口名，它就会显示出该接口的配置信息。如果调用 ifconfig时不带任何参数，它就会显示迄今为止已配置的所有接口； -a选项也将强制它显示所有非活动

```
# ifconfig eth0
eth0      Link encap 10Mbps Ethernet  HWaddr 00:00:C0:90:B3:42
          inet addr 191.72.1.2 Bcast 191.72.1.255 Mask 255.255.255.0
UP BROADCAST RUNNING  MTU 1500  Metric 0
RX packets 3136 errors 217 dropped 7 overrun 26
TX packets 1752 errors 25 dropped 0 overrun 0
```

的接口。以太网接口eth0的调用示例如下：

MTU（最大传输单元）和Metric（度量值）字段显示的是该接口当前的 MTU和度量值的值。按照惯例，度量值供某些操作系统所用，用于计算一条路由的成本。

RX和TX这两行展示了已经准备无误地收发了多少数据包、发生了多少错误、丢失了多少数据包（因内存少的原因），以及多少包是因为传输过载而遗失的。 ifconfig打印出的标本值或多或少与其命令行中的选项名相符；详情如下：

表5-1随括号中的标记名一起，列出了ifconfig能够识别的参数。选项对应的特性可以打开也可以取消，只在选项名前加一个破折号即可（ -）。

表5-1 ifconfig可以识别的参数

up	标志接口处于“ up ”状态，也就是说， IP层可以对其进行访问。这个选项用于命令行上给出一个地址之时。如果这个接口已被“ down ”选项临时性取消的话（与该选项对应的标记是UP RUNNING），还可以用于重新启用一个接口
down	标志接口处于“ down ”状态，也就是说， IP层不能对其进行访问。这个选项有效地禁止了IP通信流通这个接口。注意，它并没有自动删除利用该接口的所有路由信息。如果永久性地取消了一个接口，就应该删除这些路由条目，并在可能的情况下，提供备用路由
netmask	分配子网掩码，供接口所用。要么给一个前面是 0x的32位十六进制号码，要么采用只适用于两台主机所用的点分四段式号码。对 SLIP和PLIP接口来说，这个选项是必须配置的
pointtopoint address	该选项用于只涉及两台主机的点到点链接。对 SLIP和PLIP接口来说，这个选项是必须配置的（如果已经设置了一个点到点地址， ifconfig就会显示出POINTTOPOINT标记）
broadcast address	广播地址通常源于网络编号，通过设置主机部分的所有位得来。有的 IP采用的方案有所不同：这个选项可适用于某些奇怪的环境（如果已经设置了广播地址， inconfig就会显示出一个BROADCAST标记）
metric number	该选项可用于为接口创建的路由表分配度量值。路由信息协议（ RIP）利用度量值来构建网络路由表。ifconfig所用的默认度量值是0。如果不运行RIP程序，就没必要采用这个选项。如果要运行RIP程序，就尽量不要改变这个默认的度量值
mtu bytes	该选项用于设置最大传输单元，也就是接口一次能处理的最大字节数。对以太网接口来说， MTU的默认设置是1 500个字节；对SLIP接口来说，则是296个字节
arp	这个选项专用于以太网或包广播之类的广播网络。它启用 ARP（地址解析协议）来保护网络上各台主机的物理地址。对广播网来说，默认设置是“ on ”（开）
-arp	取消在接口上使用ARP
promisc	将接口置入promiscuous（混乱）模式。广播网中，这样将导致该接口接收所有的数据包，不管其目标是不是另一台主机。该选项允许利用包过滤器和所谓的以太网窥视技术，对网络通信进行分析。通常情况下，这对揪出网络故障的元凶来说，是相当有用的。但另一方面，如果有人蓄意攻击你的网络，也可浏览到通信数据，进而获得密码，破坏你的网络。一项重要的保证措施是杜绝任何人将他们的计算机接入你的以太网。另一个选项用于保护某些身份验证协议的安全，比如 Kerberos或SRA登录套件（该选项对应的标记是PROMISC）
-promisc	取消混乱模式
allmulti	多播地址即是向不在同一个子网上的一组主机广播数据。多播地址尚未获得内核支持（该选项对应的标记是ALLMULTI）
-allmulti	取消多播地址

5.9 netstat详解

下面，将为大家介绍一个特别有用的工具，用来检查网络配置和活动情况。它就是 netstat。

事实上，它是若干个工具的汇总

5.9.1 显示路由表

在随-r标记一起调用netstat时，将显示内核路由表，就像我们利用 route命令一样。vstout上产生的输出如下：

```
# netstat -nr
Kernel routing table
Destination     Gateway         Genmask         Flags Metric Ref Use
127.0.0.1       *               255.255.255.255 UH    1      0
191.72.1.0      *               255.255.255.0   U     1      0
191.72.2.0      191.72.1.1      255.255.255.0   UGN   1      0
```

-n选项令netstat以点分四段式的形式输出IP地址，而不是象征性的主机名和网络名。如果想避免通过网络查找地址（比如避开DNS或NIS服务器），这一点是特别有用的。

netstat输出结果中，第二列展示的是路由条目所指的网关。如果没有使用网关，就会出现一个星号（*）。第三列展示路由的概述。在为具体的 IP地址找出最恰当的路由时，内核将查看路由表内的所有条目，在对找到的路由与目标路由比较之前，将对 IP地址和genmask进行按位“与”计算。

第四列显示了不同的标记，这些标记的说明如下：

- G 路由将采用网关。
- U 准备使用的接口处于“活动”状态。
- H 通过该路由，只能抵达一台主机。例如，回送接口的路由器条目 127.0.0.1就如此。
- D 如果路由表的条目是由ICMP重定向消息生成的，就会设置这个标记（详情参见第 3章）。
- M 如果路由表条目已被ICMP重定向消息修改，就会设置这个标记。

netstat输出结果的Ref列展示了对这条路由的引用数，也就是说，有多少其他的路由（例如通过网关的路由）是建立在这条路由基础上的。最后两列展示了路由条目的使用次数和用于投递数据报的接口。

5.9.2 显示接口特性

在随-i标记一起调用时， netstat将显示网络接口的当前配置特性。除此以外，如果调用时还带上-a选项，它还将输出内核中所有接口，并不只是当前配置的接口。 vstout上，netstat的输出结果是这样的：

```
$ netstat -i
Kernel Interface table
Iface       MTU     Met     RX-OK  RX-ERR  RX-DRP  RX-OVR  TX-OK   TX-ERR  TX-DRP  TX-
10          0       0       3185   0       0       0       3185    0       0
eth0         1500   0       972633 17      20      120     628711  217     0
```

MTU和Met字段表示的是接口的 MTU和度量值值。RX和TX这两列表示的是已经准确无误地收发了多少数据包（RX-OK/TX-OK）、产生了多少错误（RX-ERR/TX-ERR）、丢弃了多少包（RX-DRP/TX-DRP），由于误差而遗失了多少包（RX-OVR/TX-OVR）。

最后一列展示的是为这个接口设置的标记。在利用 ifconfig显示接口配置时，这些标记都

采用一个字母。它们的说明如下：

- B 已经设置了一个广播地址。
- L 该接口是一个回送设备。
- M 接收所有数据包（混乱模式）。
- N 避免跟踪。
- O 在该接口上，禁用ARP。
- P 这是一个点到点链接。
- R 接口正在运行。
- U 接口处于“活动”状态。

5.9.3 显示链接

netstat支持用于显示活动或被动套接字的选项集。选项 -t、-u、-w和-x分别表示TCP、UDP、RAW和UNIX套接字连接。如果你另外还提供了一个-a标记，还会显示出等待连接（也就是说处于监听模式）的套接字。这样就可以得到一份服务器清单，当前所有运行于系统中的所有服务器都会列入其中。

在vlager上调用netstat -ta时，输出结果如下：

```
$ netstat -ta
Active Internet connections
Proto Recv-Q Send-Q Local Address    Foreign Address    (State)
tcp        0      0 *:domain         *:*                LISTEN
tcp        0      0 *:time           *:*                LISTEN
tcp        0      0 *:smtp           *:*                LISTEN
tcp        0      0 vlager:smtp      vstout:1040        ESTABLISHED
tcp        0      0 *:telnet         *:*                LISTEN
tcp        0      0 localhost:1046   vbardolino:telnet  ESTABLISHED
tcp        0      0 *:chargen        *:*                LISTEN
tcp        0      0 *:daytime        *:*                LISTEN
tcp        0      0 *:discard        *:*                LISTEN
tcp        0      0 *:echo           *:*                LISTEN
tcp        0      0 *:shell          *:*                LISTEN
tcp        0      0 *:login          *:*                LISTEN
```

上面的输出表明大多数服务器都处于等待接入连接状态。但是，第四行表明其中一条进入的连接来自 vstout，第六行表示有一条通往 vbardolino的telnet连接。利用 -a选项的话，netstat还会显示出所有的套接字。

注意 根据端口号，可以判断出一条连接是否是外出连接。对呼叫方主机来说，列出的端口号应该一直是一个整数，而对众所周知服务（well known service）端口正在使用中的被呼叫方来说，netstat采用的则是取自/etc/services文件的象征性服务名。

5.10 检查ARP表格

某些时候，查看甚至改变内核 ARP表格是非常有用的，比如说，在你怀疑某个备份的Internet地址可能会引发网络问题时更是如此。arp工具就是你的得力助手。其命令行选项如下：

```
arp [-v] [-t hwtype] -a [hostname]
arp [-v] [-t hwtype] -s hostname hwaddr
arp [-v] -d hostname [hostname...]
```

所有主机名参数都可能是象征性的主机名或采用点分四段式的 IP地址。

第一行调用显示出 IP地址或指定主机的 ARP条目，如果没有指定主机，则显示出所有已知的主机名。例如，在vlager上调用arp可能会产生下面的结果：

```
# arp -a
IP address        HW type          HW address
191.72.1.3        10Mbps 以太网    00:00:C0:5A:42:C1
191.72.1.2        10Mbps 以太网    00:00:C0:90:B3:42
191.72.2.4        10Mbps 以太网    00:00:C0:04:69:AA
```

上面的输出显示出vlager、vstout和vale的以太网地址。

利用-t选项，可只显示指定的硬件类型。显示结果可以是： ether、ax25或pronet，它们分别代表10Mbps 以太网、AMPR-AX.25和IEEE-802.5令牌环设备。

-s选项用于将主机名的以太网地址永久性地加入 ARP表格中。hwaddr参数指定的是硬件地址，默认情况下，将是以太网地址，被指定为用冒号隔开的 6个十六进制字节。当然，也可以用-t选项，为其他类型的硬件设置硬件地址。

出于某种原因，在 ARP表格内查找远程主机失败时（比如该远程主机的驱动程序漏洞百出，或网络中的另一台主机采用了它的 IP地址，使其能够冒充这台远程主机），要求大家手工把IP地址加入ARP表格。ARP表格内的硬性绑定的 IP地址也是一种能够避免以太网中别的主机冒充你的（非常有效）手段。

利用-d调用arp时，将删除与给定主机相关的所有 ARP条目。它可以用于强制接口再次尝试获取未知的IP地址。配置有误的系统中出现错误的广播 ARP信息时，这是非常有用的（不过，首先得重新配置被中断的主机）。

还可用-s选项来实施代理ARP。这是一种比较特殊的技术；通过假称两个地址指的是同一台主机（即gate）这一方式，主机（即gate）同时还扮演网关通向另一台名为 fnord的主机。这是通过出版一条fnord ARP条目来完成的，这个条目指代的是该主机自己的以太网接口。现在，主机发出ARP查找fnord时，gate将返回一条应答，其中包含它自己的以太网地址。发出查找的主机再将所有数据报发给gate，最后再由gate负责将它们转发给fnord。

透彻理解上面的描述是非常必要的，因为这种情况屡有发生：比如你想从一台 DOS版本的机器（已中断了 TCP实施，不能很好地识别路由）访问 fnord。此时使用代理 ARP，它就可能出现在DOS机器上，就像fnord位于本地子网上一样，对它而言，没有必要知道通过网关进行路由。

关于代理ARP，另一个非常重要的应用表现在你的主机之一临时性充当通向其他主机的网关时，比如通过一条拨号链接时。在前一个示例中，我已经见识过膝上型计算机 vlite，它只是偶尔通过一条PLIP链接和vlager建立连接。当然，能使这一切行之有效的前提是你准备为其提供代理ARP的主机必须和你的网关位于同一个 IP子网上。例如， vstout可以为Brewery子网（191.72.1.0）上所有主机提供代理 ARP，但对Winery子网（191.72.2.0）上的任何一台主机来说，则是不可能的。

为fnord提供代理ARP的调用结构如下；当然，所给的以太网地址必须是 gate的地址。

```
# arp -s fnord 00:00:c0:al:42 e0 pub
```

再通过调用的方式，删除代理ARP条目：

```
# arp -d fnord
```

5.11 未来展望

连网技术不断在进步。内核层的主要变动给我们带来了更为灵活的配置方案，使我们能够在运行时配置网络设备。例如， ifconfig命令可以采用设置IRQ和DMA通道的参数。

另一个即将面世的主要变动是在route命令中增加了mtu标记，这样就可以为特定路由设置“最大传输单元”了。最大传输单元简称 MTU。路由专用的MTU优先于为接口指定的 MTU。人们一般都会将这个选项用于通过网关的路由，这种情况下，网关和目标主机之间的链接需要非常低的MTU值。打个比方，假设主机wanderer通过一条SLIP链接与vlager相连。在把数据从vstout发给wanderer时，wanderer上的网络层将采用1 500字节（最大值）的标准来打包数据，因为是通过以太网来发送数据。而另一方面， SLIP链接只能传送MTU值为296的数据包，所以vlager上的网络层不得不将IP包分成296个字节大小的数据段。如果不这样做，就要对 vstout上的路由进行重新配置，使其一开始就采用 296的MTU值，这样可避免花销太大的数据分段：

```
# route add wanderer gw vlager mtu 296
```

注意， mtu选项还允许你选择性地取消“子网属本地网络”策略（ SNARL）的影响。这个策略是一个内核配置选项，我们曾在第 3章讨论过。

5.12 名字服务和解析器配置

正如第2章描述的那样， TCP/IP连网可能依赖于某些不同的方案，将主机名转换为相应的地址。一种最简单的方法便是主机表，它没有采用类似于把域名空间分成若干个区的方式，这个主机表保存在 /etc/hosts文件内。但它只适用于小型的局域网，这些局域网由一个管理员负责。主机文件的格式参见第 3章。

另外，也可以用BIND即Berkeley Internet域名服务，将主机名解析为 IP地址。BIND的配置相当琐碎，但一旦配置好了，要在网络拓扑中进行更改就变得非常简单。在许多 Linux系统上，域名服务是通过一个名为 named的程序来提供的。该程序在启动时，将 master（主要）文件载入自己的缓存内，等待远程或本地用户进程的查询。设置 BIND的方法很多，而且都不要求你在各台主机上运行域名服务器。

本章将简要介绍一下如何操控域名服务器。如果你手里的网络不仅仅是一个小型局域网，而且还有可能接入因特网，那么，要想在这种环境下使用 BIND，最好找一本关于 BIND的书来作参考。推荐大家使用 Cricket Liu所著的《DNS和BIND》，该书由O’Reilly出版，现在已有了第三版，可访问 www.oreilly.com/catalog/dns3/获得你需要的资料。关于这方面的最新信息，还可查看一下 BIND源代码中包含的注意事项。另外，还有两大新闻组供大家参考：comp.protocols.dns和comp.protocols.dns.bind。

5.13 解析器库

“解析器”并不是一种特殊的应用程序，而是解析器库的代名词，它是一个可在标准 C语言库内能够找到的函数集合。核心例程分别是： gethostname(2)和gethostbyaddr(2)，这两个例

程用于查找主机对应的所有IP地址，反之亦然。它们还可以配置为简单地查询主机内的信息，查询域名服务器或使用NIS（网络信息服务）的hosts（主机）数据库。其他应用程序，比如smail，可能包含不同的驱动程序，需要特别注意。

5.13.1 host.conf文件

控制解析器设置的核心文件是host.conf。它驻留在/etc中，并告诉解析器使用哪些服务，使用顺序又如何。

host.conf文件内的选项必须以单行的形式出现。各个字段必须用空格隔开。“#”号引入的是对下一行进行扩展的批注。

下列选项是可用的：

■ order —— 该选项决定解析服务的使用顺序。有效选项有：

- bind —— 用于查询域名服务器。
- hosts —— 用于/etc/hosts内的查找。
- nis —— 用于NIS查找。

可以同时指定一个选项或所有的选项。各选项在命令行出现的顺序将决定各自的服务顺序。

■ multi —— 其选项有on和off。它用于判断/etc/hosts内的主机是否获得允许使用多个IP地址，如果可以，这个主机就被称为“多宿主主机”。这个标记对DNS和NIS查询是没有用的。

■ nospoof —— 就像前一章解释的那样，DNS允许你利用inaddr.arpa域，查找属于某一台主机的主机名。域名服务器提供假主机名的作法称为“电子欺骗”。为防止这种情况的发生，可将解析器配置为查看原始IP地址是否真的与获得的主机名相符。如果不相符，这个主机名就会被丢弃并返回一个错误。将“电子欺骗”设为“on”，就可以启用这个行为。

■ alert —— 该选项将on和off作为其参数。如果设置为“on”，任何电子欺骗尝试（如上所述）将令解析器将消息记录到系统日志中。

■ trim —— 该选项将域名作为其参数，在实施主机名查找之前，这里的域名会从主机名文件中被删除。这个特性对主机条目来说，是非常有用的，因为人们只想指定不带域名的主机名。对绑定本地域名的主机所进行的查找将删除这个特性，因此，允许/etc/hosts文件内的查找继续下去。trim选项累积起来，使其很容易将你的主机视为处于本地域内。

vlager所用的文件范例如下所示：

```
# /etc/host.conf
# We have named running, but no NIS (yet)
order    bind hosts
# Allow multiple addrs
multi    on
# Guard against spoof attempts
nospoof on
# Trim local domain (not really necessary).
trim     vbrew.com.
```

5.13.2 解析器环境变量

利用大量的环境变量，可以改写host.conf文件中的设置。

它们是：

■ RESOLV HOST CONF —— 该变量指定准备读取的文件，并用该文件来代替/etc/host.conf。

■ RESOLV SERV ORDER —— 改写host.conf文件内指定的order（顺序）选项。指定的服务分别为hosts、bind和nis，并分别用空格、逗号、冒号和分号隔开。

■ RESOLV SPOOF CHECK —— 判断针对电子欺骗的尺度。如果设置为 off，则表示彻底取消。warn和warn off这两个值都会启用电子欺骗校验，但将分别打开和关闭记录。“ * ”值将打开电子欺骗校验，但会像host.conf文件内定义的那样，保留记录设备。

■ RESOLV MULTI —— 可选项有on和off，用于改写tt host.conf文件内的multi选项。

■ RESOLV OVERRIDE TRIM DOMAINS —— 该环境变量指定一份trim域清单，它们将用于改写host.conf内的那些域。

■ RESOLV ADD TRIM DOMAINS —— 该环境变量指定一份trim域清单，这些域将增加到host.config中指定的域内。

5.13.3 域名服务器查找 —— resolv.con的配置

在配置解析器库，使之利用 BIND域名服务进行主机查找时，还必须告诉它准备使用哪些域名服务器。对此，有一个专门的文件，名为 resolv.conf。如果该文件不存在，或者为空，解析器就会假定域名服务器在你的本地主机上。

如果在本地主机上运行一个域名服务器，就必须单独安装它，具体做法参见下一小节。如果你处于一个本地网络上，而且有机会采用一个现成的域名服务器，用这个现成的最好。

resolv.conf文件中，最重要的选项是nameserver（域名服务器），它将给出准备使用的域名服务器的IP地址。如果你通过nameserver选项，多次指定了若干个域名服务器，查找就会按照指定时的顺序来进行。因此，应该将最稳定可靠的服务器放在最前面。当前，最多可指定 3个服务器。

如果不采用nameserver选项，解析器就会试着链接到本地主机上的域名服务器。

另外两个选项：domain和search，用于处理绑定在主机名上的默认域（前提是 BIND在第一条查询中未能解析出这个主机名。 search选项指定了一份准备查找的域名清单。清单中的项目用空格隔开。

如果不采用search选项，解析器就会利用域名本身，再加上上推至根的所有父域名，根据本地域名构建一份默认查找清单。本地域名的赋予是通过 domain（域）语句来进行的；如果一个选项也不指定，解析器就会通过 getdomainname(2)系统调用，将域名包括在内。

如果你对此仍然不很清楚，可仔细想想 Virtual Brewery所用的resolv.conf文件，如下所示：

```
# /etc/resolv.conf
# Our domain
domain          vbrew.com
#
# We use vlager as central nameserver:
nameserver      191.72.1.1
```

在解析主机名 vale时，解析器将查找 vale，如果失败，就会查找 vale.vbrew.com和vale.com。

5.13.4 解析器的健壮性

如果在一个大型网内运行一个局域网，在可能的情况下，人们肯定会采用中央域名服务器。其好处在于能够开发出这些服务器的巨大潜力，因为所有的查询都将转发到这些中央服务器来。但是，这一方案有个明显的不足：校园网内的主干线发生突发性的灾难事件时，各系的局域网也将无法正常运行，因为解析器不能抵达任何一台中央域名服务器。也无法从任何一台X终端登录，不能打印，什么也不能做。

尽管此类事件并不是频频发生，但最好能做到防患于未然。

一种办法是设置一个本机域名服务器，解析你所处的本地域内的主机名，并将所有对其他主机名的查询转发给主服务器。当然，前提是你在自己的域内运行。

另一种办法是：把自己的域或局域网备份主机表保存在 /etc/hosts文件内。然后，再将“ order bind hosts ”主机包括在/etc/hosts.conf内，在中央域名服务器关闭时，令解析器退回主机文件，继续查找。

5.14 named的运行

在大多数计算机上提供域名服务器的程序通常被命名为 named。这个服务器程序起初是为BSD开发的，用于为客户机提供域名服务，还可用于其他的域名服务器。当前，大多数人安装的版本是BIND 8.2（8.X是源于4.X系列的下一个主版本；目前4.x的版本号是4.9.7）。

注意 BIND的维护工作由“ 因特网软件协会 ”（Internet Software Consortium，其网址是www.isc.org）负责。大家可在www.isc.org/bind.html上找到需要的BIND资料。

本小节要求大家掌握域名系统（ DNS）的工作原理。如果觉得下面的讨论如同天书，就应该回头看看第2章，了解DNS的基本知识。

named的启动通常是在系统启动时开始的，而且会一直处于运行状态直到关机。它从一个名为/etc/named.boot的配置文件中，取出自己的有关信息，而另外一些文件中包含将域名映射为地址的所有数据。后者叫作“ zone ”（区）文件。这类文件的格式和含义将在下一小节讨论。

要运行named，只须在提示行键入

```
# /usr/sbin/named
```

然后， named出现，并读取named.boot文件和所有指定的 zone文件。再以ASCII的形式，将自己的进程ID写入/var/run/named.pid，如有必要，将从主服务器下载所有的 zone文件，并在端口53上，开始监听DNS查询。

注意 目前，许多FTP站点上都有named二进制文件，各自的配置稍有不同。有的将自己的pid文件放在/etc内，有的则保存在/tmp或/var/tmp内。

5.14.1 named.boot文件

named.boot文件一般都非常小，其中包含一些指向 master文件和其他域名服务器的指针，master文件中包含的是区信息。启动文件中的批注以分号开头，并扩展到下一行。在详细讨论named.boot文件的格式之前，我们将看看 vlager主机的示范文件：

```
;
; /etc/named.boot file for vlager.vbrew.com
;
directory      /var/named
;
;              domain                    file
;-------------------------------------
cache          .                         named.ca
primary        vbrew.com                 named.hosts
primary        0.0.127.in-addr.arpa      named.local
primary        72.191.in-addr.arpa       named.rev
```

上面的示例中，cache和primary命令将信息载入named文件。该信息取自第二个参数中指定的master文件。其中包含DNS源记录的文本描述，详情如下。

这个示例中，我们将named配置为三个域的primary（主要）域名服务器，如同文件最后primary语句指定的那样。比如说，该示例中的第一行从名为 named.hosts的文件中，取出区数据，指令named充当vbrew.com的主要域名服务器。 directory（目录）关键字告诉它所有的区文件都在/var/named目录下。

cache条目非常特殊，其作用包括两方面：指导 named激活自己的缓冲区，并载入 cache文件指定的根域名服务器提示（在我们的示例中是 named.ca）。稍后将为大家详细讨论域名服务器提示。

下面列出了named.boot文件中最重要的几个选项：

■ diretory —— 该选项指定区文件驻留的目录。另外，还可能给出与该目录相关的文件名。重复利用 directory选项，可指定若干个目录。按照 Linux文件系统标准，这个目录应该是/var/named。

■ primary —— 该选项将一个域名和一个文件名作为自己的参数，声明本地域名服务器属于named域。作为一个主要域名服务器， named从给定的master文件加载区信息。一般说来，每个root文件中，至少总都应该有一个主要条目，用于逆向映射网络 127.0.0.0，该网络就是本地回送网络。

■ secondary —— 该选项将域名、地址清单和文件名作为自己的参数。它声明本地域名服务器是指定域的辅助主要服务器。辅助服务器中也保存有该域名的管理数据，但它不会从不同的文件收集这些数据，而是试图从主要服务器上下载。因此，地址清单中，至少应该为named指定一个主要服务器的IP地址。然后，本地服务器将依次和各域名服务器取得联系，直到成功传送区数据库，然后，这个区数据库便保存在第三个参数中指定的备份文件中。如果无主要域名服务器作出应答，本地服务器只好从备份文件中获取区数据了。最后， named将定期对区数据进行更新。详情可参见 SOA源记录类型。

■ cache —— 该选项将域名和文件名作为自己的参数。这个文件中包含根服务器提示，是一份指向根域名服务器的记录清单。目前，只能识别 NS和A类型的记录。域名参数一般是根域名“ .”。这一点对named而言是绝对重要的：如果cache选项不能在根文件中执行， named就根本不能开发出一个本地缓冲区。如果下一个服务器查询不是在本地网络上进行，必然会降低性能，增加网络负担。更有甚者， named将不能抵达任何一台根域名服务器，因而也不可能解析自己管理范围之外的其他地址。但在使用转发服务器时，情况有所不同（比较下面的forwarders选项）。

■ forwarders —— 该选项将地址清单作为自己的参数。该清单中的 IP地址指定的是一系列的域名服务器，如果named不能从自己的本地缓冲区内解析地址查询，它就会要求查询清单中的这些域名服务器。named依次对这些服务器进行查询，直到其中之一能够作出应答。

■ slave —— 该选项令域名服务器作为从属服务器。也就是说，这类服务器自身永远不会执行递归查询，只是将查询转发到 forwarders中指定的服务器。

sortlist和domain两个选项不在这里的讨论之列。另外，区数据库文件内，还有两条指令，它们是$INCLUDE和$ORIGIN。由于平常较少用到它们，所以，我们也不打算深入讨论。

5.14.2 DNS数据库文件

master文件包含在named.hosts之类的named文件内， master文件一直都有一个与之关联的域，叫作origin。这个域是用 cache和primary命令指定的。 master文件内部，允许你指定和该域相关的域和主机名。如果配置文件中给定的域名最后是一个句点“ .”，就会被视为绝对名，否则就被视为与origin相关。用“ @ ”，可借代这个origin。

包含在master文件内的所有数据被分为若干条 resource records（源记录，简称RR）。它们组成能通过DNS的最小信息单元。每条源记录都有类型。比如说 A类型的记录，将主机名映射为IP地址，而CNAME类型的记录，则将主机的别名和其正式主机名关联在一起。示例可参见清单5-1，该示例将向大家展示virtual brewery网络的named.hosts master文件。

master文件内的源记录表达式的格式如下：

```
[domain] [ttl] [class] type rdata
```

每个字段间用空格隔开。如果新行以大括号 {开头，最后一个字段以大括号 }结尾，这个条目就会在以后的几行内重复地出现。冒号和新行之间的所有记录都会被忽略。

■ domain —— 指定条目所用的域名。如果不指定域名， RR就沿用上一条RR采用的域。

■ ttl —— 为了强制解析器定期丢弃信息，每个 RR都有一个“ 生存时间 ”（ttl）。ttl字段指定自服务器获取的信息之有效时间，以秒计数。它是一个十进制数，最多有 8个数位。如果不指定ttl值，就会沿用前一条SOA记录的最小值。

■ class —— 这是一个地址类，类似于 IP地址的IN或Hesiod类中的对象所用的 HS。对TCP/IP连网来说，这个字段必须是IN。如果不指定类字段，将沿用前一条 RR的类。

■ type —— 该字段描述RR的类型。最常用的类型是 A、OA、PTR和NS。后续的小节将对RR的不同类型进行深入讨论。

■ rdata —— 该字段中容纳的是与RR关联的数据。它的格式与RR的类型有关。下面将针对各种RR类型，解释这一字段的格式。

下面列举了DNS master文件内所用的RR，但不完整。另外两条不常用的 RR将不作讨论。

■ SOA —— 描述权限区（SOA的意思是“ start of authority ”）。它标志SOA RR之后的记录中包含这个域的管理信息。 primary选项中包含的master文件内必须包含该权限区的 SOA记录。源数据中包含下面的字段：

■ origin —— 这是该域之主要域名服务器的规范主机名。通常被指定为绝对主机名。

■ contact —— 负责维护一个域的人的电子邮件地址，不过句点（ .）代替了“ @ ”。例如，Virtual Brewery的负责人是janet，那么这个字段就应该是janet.vbrew.com。

■ serial —— 区文件的版本号，用一个单独的十进制数来表示。只要区文件内的数据发生

了变化，这个数也应该相应增加。序列号是辅助域名服务器用以识别区信息何时发生变化的根据。为了保持更新，辅助服务器定期请求主要服务器的 SOA记录，并把它的序列号和缓存起来的SOA记录的序列号进行比较。如果不同，辅助服务器就会从主要服务器内将整个区数据库传回来。

■ refresh —— 指定时间间隔，以秒计。表示辅助服务器在两次复查主要服务器的 SOA记录期间，应该等待多久。再次提醒大家注意，这也是一个十进制数，最大有 8个数位。通常情况下，网络结构是不会经常变化的，所以对大型网来说，这个间隔可指定为一天，对小型网来说，甚至可以更长。

■ retry —— 指定时间间隔，表示在请求或区更新失败的情况下，辅助服务器应该隔多久才能重新尝试与主要服务器联系。这个值不能太低，因为服务器的临时性故障或网络问题可能导致辅助服务器浪费网络资源。最好设为 1.5h。

■ expire —— 指定时间，以秒计。这段时间一过，如果服务器仍然不能与主要服务器沟通，它就会将全部区数据最终丢弃。这个值一般较大。 Craig Hunt建议值是42天。

■ minimum —— 默认的ttl值，用于没有显示指定生存时间的源记录。它要求其他域名服务器在间隔一段时间后，丢弃 RR。但是，这个值和辅助服务器试着更新区信息的时间间隔是不相干的。这个minimum应该是一个非常大的值，对网络结构几乎一成不变的局域网来说，更是如此。最好设为一周或一个月。如果个别 RR可能频繁更动，仍然可以单独为它们设置 ttl。

■ A —— 将一个IP地址和主机名关联在一起。源数据字段中包含的地址是点分四段式。每台主机必须只有一条 A记录。这条A记录中的主机名被视为正式主机名或规范主机名。其他主机名均为别名，而且必须利用一条 CNAME记录，将它们与规范主机名对应。

■ NS —— 指向下属区的主（master）域名服务器。含有NS记录的原因，参见第3章。源数据字段中包含该域名服务器的主机名。要对这个主机名进行解析，就需要 A记录，它就是指定域名服务器之IP地址的所谓的glue record。

■ CNAME —— 将主机的别名和其规范主机名关联在一起。规范主机名是 master文件为一条A记录提供的；别名只是通过一条 CNAME记录，和规范主机名链接起来，而不是真正地拥有记录。

■ PTR —— 这类记录用于将 inaddr.arpa域内的名字和主机名关联起来。用于将 IP地址逆向映射为主机名，但所给的主机名必须是规范主机名。

■ MX —— 该RR宣布域的邮件交换器。宣布邮件交换器的原因，可参见第 14章。MX记录的格式如下：

```
[domain] [ttl] [class] MX preference host
```

host对域邮件交换器进行了命名。每个邮件交换器都有一个相关的整数型首选项值。对负责将邮件投递到域的邮件传输代理来说，它们将试着将邮件投递到拥有该域 MX记录的所有主机，直到成功为止。先从首选项值低的着手，从低到高尝试所有的主机。

■ HINFO —— 该记录提供系统硬件和软件的相关信息。其格式如下：

```
[domain] [ttl] [class] HINFO hardware software
```

hardware字段标识该主机所用的硬件。关于这个字段的指定，有相关的约定。其有效值可参考“ 已分配编号 ”（RFC 1340）。如果该字段中有空格，则必须用双引号将其封闭起来。software字段指的是该主机所用的操作系统软件。其有效值同样可从“ 已分配编号 ”（RFC

1340）中获得。

5.14.3 编写Master文件

清单5-1、5-2、5-3和5-4给出了关于域名服务器的几个示范文件，这些文件用于 brewery网络中的vlager主机。由于已经介绍过这个网络的特性（是一个单一的局域网），所以这里的示例相当简单。如果你面对的网络较为复杂，就不能只运行 named，还应参考Cricket Liu和Paul Albitz合著的《DNS和BIND》一书。

清单5-1中的named.ca cache文件展示了根域名服务器的提示记录示例。一个典型的 cache文件内，一般都描述了一打以上的域名服务器。利用本章最后要讲的 nslookup，便可获得根域的所有当前域名服务器清单。

注意 在没有安装根服务器提示：catch-22的情况下，是不能在自己的域名服务器内查询根服务器的！要避免此类情况的发生，可令nslookup另行采用域名服务器，或从清单5-1中的示范文件着手，获得一份列有所有有效服务器的清单。

清单5-1 named.ca文件

```
;
; /var/named/named.ca          Cache file for the brewery.
;                 We're not on the Internet, so we don't need
;                 any root servers. To activate these
;                 records, remove the semicolons.
;
; .                 99999999    IN    NS   NS.NIC.DDN.MIL
; NS.NIC.DDN.MIL    99999999    IN    A    26.3.0.103
; .                 99999999    IN    NS   NS.NASA.GOV
; NS.NASA.GOV       99999999    IN    A    128.102.16.10
```

清单5-2 named.hosts文件

```
;
; /var/named/named.hosts       Local hosts at the brewery
;                               Origin is vbrew.com
;
@                   IN  SOA    vlager.vbrew.com. (
                               janet.vbrew.com.
                               16          ; serial
                               86400       ; refresh: once per day
                               3600        ; retry:   one hour
                               3600000     ; expire:  42 days
                               604800      ; minimum: 1 week
                               )
                    IN  NS     vlager.vbrew.com.
;
; local mail is distributed on vlager
                    IN  MX     10 vlager
;
; loopback address
localhost.          IN  A      127.0.0.1
; brewery Ethernet
```

```
vlager              IN  A      191.72.1.1
vlager-if1          IN  CNAME  vlager
; vlager is also news server
news                IN  CNAME  vlager
vstout              IN  A      191.72.1.2
vale                IN  A      191.72.1.3
; winery Ethernet
vlager-if2          IN  A      191.72.2.1
vbardolino          IN  A      191.72.2.2
vchianti            IN  A      191.72.2.3
vbeaujolais         IN  A      191.72.2.4
```

清单5-3 named.local文件

```
;
; /var/named/named.local        Reverse mapping of 127.0.0
;                                Origin is 0.0.127.in-addr.arpa.
;
@                   IN  SOA    vlager.vbrew.com. (
                               joe.vbrew.com.
                               1           ; serial
                               360000      ; refresh: 100 hrs
                               3600        ; retry:   one hour
                               3600000     ; expire:  42 days
                               360000      ; minimum: 100 hrs
                               )
                    IN  NS     vlager.vbrew.com.
1                   IN  PTR    localhost.
```

清单5-4 named.rev文件

```
;
; /var/named/named.rev          Reverse mapping of our IP addresses
;                                Origin is 72.191.in-addr.arpa.
;
@                   IN  SOA    vlager.vbrew.com. (
                               joe.vbrew.com.
                               16          ; serial
                               86400       ; refresh: once per day
                               3600        ; retry:   one hour
                               3600000     ; expire:  42 days
                               604800      ; minimum: 1 week
                               )
                    IN  NS     vlager.vbrew.com.
; brewery
1.1                 IN  PTR    vlager.vbrew.com.
2.1                 IN  PTR    vstout.vbrew.com.
3.1                 IN  PTR    vale.vbrew.com.
; winery
1.2                 IN  PTR    vlager-if1.vbrew.com.
2.2                 IN  PTR    vbardolino.vbrew.com.
3.2                 IN  PTR    vchianti.vbrew.com.
4.2                 IN  PTR    vbeaujolais.vbrew.com.
```

5.14.4 验证域名服务器的设置

这里有一个很优秀的工具，可用于验证域名服务器的设置。其名为 nslookup，既可以交互式操作，又可以通过命令行操作。如果采用后一种方式，只须这样调用它：

```
nslookup hostname
```

然后，它就开始在resolv.conf指定的域名服务器查询主机名（如果改文件命名的服务器不止一个，nslookup将依次进行查找）。

但在交互模式中，情形更为有趣。除了查找个别的主机外，还要查找各种类型的 DNS记录并传送域的整个区信息。

如果调用nslookup时不带参数，nslookup就会显示出它所用的域名服务器，并进入交互式模式。出现“ > ”提示时，可键入任何一个你认为应该查询的域名。默认情况下，它会要求 A类型的记录，即其中包含与该域名相关的IP地址。

当然，也可执行“ set type=type ”，改变这个类型，后一个 type应该是上面所讲的源记录名之一或ANY。

比如，和nslookup的对话可能如下所示：

```
$ nslookup
Default Name Server:  rs10.hrz.th-darmstadt.de
Address:  130.83.56.60

 sunsite.unc.edu
Name Server:  rs10.hrz.th-darmstadt.de
Address:  130.83.56.60

Non-authoritative answer:
Name:    sunsite.unc.edu
Address:  152.2.22.81
```

如果是查询这样的域名 —— 无相关 IP地址、但可在 DNS数据库内找到其别的记录时，nslookup将无功而返，并带回这样的错误消息：“ No type A records found ”（未找到A类记录）。但是，可以利用set type命令，查询其他类型的记录。比如，要想获得 unc.edu的SOA，可以执行下列程序：

```
 unc.edu
*** No address (A) records available for unc.edu
Name Server:  rs10.hrz.th-darmstadt.de
Address:  130.83.56.60

 set type=SOA
 unc.edu
Name Server:  rs10.hrz.th-darmstadt.de
Address:  130.83.56.60

Non-authoritative answer:
unc.edu
        origin = ns.unc.edu
        mail addr = shava.ns.unc.edu
        serial = 930408
```

```
        refresh = 28800 (8 hours)
        retry   = 3600 (1 hour)
        expire  = 1209600 (14 days)
        minimum ttl = 86400 (1 day)

Authoritative answers can be found from:
UNC.EDU nameserver = SAMBA.ACS.UNC.EDU
SAMBA.ACS.UNC.EDU       internet address = 128.109.157.30
```

类似的方式还可用于查询MX记录等。使用ANY类型时，将返回与指定名相关的所有源记录。

```
 set type=MX
 unc.edu
Non-authoritative answer:
unc.edu preference = 10, mail exchanger = lambada.oit.unc.edu
lambada.oit.unc.edu     internet address = 152.2.22.80

Authoritative answers can be found from:
UNC.EDU nameserver = SAMBA.ACS.UNC.EDU
SAMBA.ACS.UNC.EDU       internet address = 128.109.157.30
```

除了用于调试设置之外，nslookup还可实际用于获得named.ca文件之当前根域名服务器列表。这是查询与根域关联的所有NS类型的记录而得的。

```
 set typ=NS

Name Server:  fb0430.mathematik.th-darmstadt.de
Address:  130.83.2.30

Non-authoritative answer:
(root)  nameserver = NS.INTERNIC.NET
(root)  nameserver = AOS.ARL.ARMY.MIL
(root)  nameserver = C.NYSER.NET
(root)  nameserver = TERP.UMD.EDU
(root)  nameserver = NS.NASA.GOV
(root)  nameserver = NIC.NORDU.NET
(root)  nameserver = NS.NIC.DDN.MIL

Authoritative answers can be found from:
(root)  nameserver = NS.INTERNIC.NET
(root)  nameserver = AOS.ARL.ARMY.MIL
(root)  nameserver = C.NYSER.NET
(root)  nameserver = TERP.UMD.EDU
(root)  nameserver = NS.NASA.GOV
(root)  nameserver = NIC.NORDU.NET
(root)  nameserver = NS.NIC.DDN.MIL
NS.INTERNIC.NET internet address = 198.41.0.4
AOS.ARL.ARMY.MIL        internet address = 128.63.4.82
AOS.ARL.ARMY.MIL        internet address = 192.5.25.82
AOS.ARL.ARMY.MIL        internet address = 26.3.0.29
C.NYSER.NET     internet address = 192.33.4.12
TERP.UMD.EDU    internet address = 128.8.10.90
NS.NASA.GOV     internet address = 128.102.16.10
```

```
NS.NASA.GOV      internet address = 192.52.195.10
NS.NASA.GOV      internet address = 45.13.10.121
NIC.NORDU.NET    internet address = 192.36.148.17
NS.NIC.DDN.MIL   internet address = 192.112.36.4
```

利用nslookup内部的help命令，便可获得nslookup能够使用的所有命令。

5.14.5 其他工具

还有几个工具，有助于你成为更好的 BIND管理员。下面简要为大家介绍两个。具体用法参见它们的相关文档。

hostcvt这个工具对最初的 BIND配置来说，是非常有帮助的，具体说来，是将你的/etc/hosts文件转换为用于 named的master文件。它同时生成了 forward（A类）和逆向映射（PTR类）条目，要注意别名等等。虽然它不可能全盘接管你的任务，比如你打算自行调节SOA记录中的超时值，增加MX条目等等，至少说来，能为你省几片阿斯匹林（意为为你省心）。hostcvt是BIND源代码中的一部分，少数FTP站点上也有其独立封装版本。

面对已设置好的域名服务器，你肯定想迫不及待地测试一番。据我所知，目前唯一的理想工具是dnswalk，它引导着你步步深入DNS数据库，查找常见错误，验证其间的信息是否一致。dnswalk新近已在comp.sources.misc上发布。归入DNS组的FTP站点都能使用它（如果你对这类站点一无所知，到ftp.uu.net去，准没错）。